**Final Project**

I chose the Amazon Musical Instrument Data set for the raw data for my final project. This raw data is available from <https://www.kaggle.com/eswarchandt/amazon-music-reviews> . I also found a back up dataset from the University of California San Diego <http://jmcauley.ucsd.edu/data/amazon> . I also found additional pay for information sites at such as <https://www.musictrades.com/census.html> and <https://www.statista.com/outlook/19020000/109/musical-instruments/united-states> . However, I did not go so far as to purchase additional data from any other sources at this time. The data file came as a Zip file from Kaggle and would not read in until I unzipped the file.

The data file itself seemed to be pretty clean when I reviewed it and did not make any modifications prior to uploading the file. The data is in a JSON file with nine columns and over ten thousand rows of data from reviews given for musical instrument purchases during 2013-2014.

My program utilized a Jupyter notebook in Anaconda from the importation of the JSON file into a Pandas dataframe from a file that was unzipped and stored on my computer hard drive in the programs file. The Pandas dataframe consisted of nine columns and over 10,260 rows of data from Amazon’s data base of reviews for musical instruments. The majority of these columns were Amazon customer information that was alpha numeric and provided no real benefit to the analysis, such as customer ID number, asin number, and helpfulness rating. I then utilized the filter function to cull columns down to the most meaningful two columns, Customer Overall rating of the transaction and the customers actual review text. I kept the overall rating in the dataset and ran the NLTK Sentiment analyzer on the data which resulted in returning a neutral sentiment rating. Then, I removed the Overall rating from the dataset with the data frame filter function again.

Next, I attempted to put the ‘reviewText’ data into a list in order to Tokenize the data.

Finally, I ran the NLTK Recursive Descent Parser utilizing a grammar to search for keywords in the dataset to include Gibson Guitars and guitars in general. I had hoped the resulting output would help me to answer the question; What are the sales of highly reviewed and regarded guitar brands like Gibson on Amazon? This particular question was quite revised from the Proposal for the Final Project due to the much more specific question I had formulated in order to understand what brought Gibson Guitars to file for bankruptcy in 2018.

I was not happy with the overall output of the program since I really could not get the parser to parse any of the data beyond the header of the dataframe. I suppose this could be in part due to the grammar I am using or the manner in which I am using the parsing tool.

This is of course my first class or project utilizing Python, so it could quite assuredly be my own personal inexperience.

The basic summary for the description of my programs is as follows:

* Data sourced from Amazon - JSON data file
* Pandas dataframe
* Sentiment analysis
* Tokenization
* Parsing

The project was solely performed and theorized by myself, without the help of others in our class or otherwise. It is my own original work based on my personal interest as a guitar player, maker, and collector as to the reasoning behind the recent decline of a major maker in the industry. In hindsight, I could see that there are some improvements that I could make to the overall design of the project and that would be to use more focused data surrounding the 2018 bankruptcy of Gibson, Gibson manufacturing data and sales data, or some form of pay data that I has previously found on sites like Statista. I know from being in the industry fringes, that peoples’ general sentiment is that Gibson’s overall ‘quality’ has gone down tremendously over the last 5 years or so. Also, I was capable of looking through the raw data file well enough to say that there was not much pertaining to guitars that were purchased from Amazon, but a lot of accessories and guitar effects pedals. I did not find anything specifically about Gibson.

In conclusion, Amazon is the shopping place of a lot of Americans, however, the 2013 – 2014 data for musical instruments from Amazon does not shed much light on people’s preferences for guitars in the online marketplace especially for specific information for Gibson Guitars. My overall topic of interest is probably too specific for this one dataset. I would need to look into the individual guitar makers sales data to better determine some of the answer my question of what has caused the downturn in sales for Gibson causing their 2018 bankruptcy.

**Code & Output**

*# This Python 3 environment comes with many helpful analytics libraries installed*

*# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python*

*# For example, here's several helpful packages to load*

**import** **os**

**import** **json**

**import** **nltk**

**import** **re**

**import** **urllib.request**

**import** **numpy** **as** **np** *# linear algebra*

**import** **pandas** **as** **pd** *# data processing, CSV file I/O (e.g. pd.read\_csv)*

**from** **pandas** **import** read\_json

*# import data from Kaggle.com - but will not load a zip file - "https://www.kaggle.com/eswarchandt/amazon-music-reviews?select=Musical\_Instruments\_5.json"*

**for** dirname, \_, filenames **in** os.walk('/kaggle/input'):

**for** filename **in** filenames:

print(os.path.join(programsPKA, Musical\_Instruments\_5.json))

In [4]:

*# define the pandas dataframe and call the extracted json data file from files; filter and parse data down to dataset*

df=pd.read\_json("Musical\_Instruments\_5.json",lines=**True**)

dataset=df.filter(['overall','reviewText'],axis=1)

df

Out[4]:

|  | **reviewerID** | **asin** | **reviewerName** | **helpful** | **reviewText** | **overall** | **summary** | **unixReviewTime** | **reviewTime** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | A2IBPI20UZIR0U | 1384719342 | cassandra tu "Yeah, well, that's just like, u... | [0, 0] | Not much to write about here, but it does exac... | 5 | good | 1393545600 | 02 28, 2014 |
| **1** | A14VAT5EAX3D9S | 1384719342 | Jake | [13, 14] | The product does exactly as it should and is q... | 5 | Jake | 1363392000 | 03 16, 2013 |
| **2** | A195EZSQDW3E21 | 1384719342 | Rick Bennette "Rick Bennette" | [1, 1] | The primary job of this device is to block the... | 5 | It Does The Job Well | 1377648000 | 08 28, 2013 |
| **3** | A2C00NNG1ZQQG2 | 1384719342 | RustyBill "Sunday Rocker" | [0, 0] | Nice windscreen protects my MXL mic and preven... | 5 | GOOD WINDSCREEN FOR THE MONEY | 1392336000 | 02 14, 2014 |
| **4** | A94QU4C90B1AX | 1384719342 | SEAN MASLANKA | [0, 0] | This pop filter is great. It looks and perform... | 5 | No more pops when I record my vocals. | 1392940800 | 02 21, 2014 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **10256** | A14B2YH83ZXMPP | B00JBIVXGC | Lonnie M. Adams | [0, 0] | Great, just as expected. Thank to all. | 5 | Five Stars | 1405814400 | 07 20, 2014 |
| **10257** | A1RPTVW5VEOSI | B00JBIVXGC | Michael J. Edelman | [0, 0] | I've been thinking about trying the Nanoweb st... | 5 | Long life, and for some players, a good econom... | 1404259200 | 07 2, 2014 |
| **10258** | AWCJ12KBO5VII | B00JBIVXGC | Michael L. Knapp | [0, 0] | I have tried coated strings in the past ( incl... | 4 | Good for coated. | 1405987200 | 07 22, 2014 |
| **10259** | A2Z7S8B5U4PAKJ | B00JBIVXGC | Rick Langdon "Scriptor" | [0, 0] | Well, MADE by Elixir and DEVELOPED with Taylor... | 4 | Taylor Made | 1404172800 | 07 1, 2014 |
| **10260** | A2WA8TDCTGUADI | B00JBIVXGC | TheTerrorBeyond | [0, 0] | These strings are really quite good, but I wou... | 4 | These strings are really quite good, but I wou... | 1405468800 | 07 16, 2014 |

10261 rows × 9 columns

In [5]:

*# display the dataset for "overall" & "reviewText"*

dataset

Out[5]:

|  | **overall** | **reviewText** |
| --- | --- | --- |
| **0** | 5 | Not much to write about here, but it does exac... |
| **1** | 5 | The product does exactly as it should and is q... |
| **2** | 5 | The primary job of this device is to block the... |
| **3** | 5 | Nice windscreen protects my MXL mic and preven... |
| **4** | 5 | This pop filter is great. It looks and perform... |
| **...** | ... | ... |
| **10256** | 5 | Great, just as expected. Thank to all. |
| **10257** | 5 | I've been thinking about trying the Nanoweb st... |
| **10258** | 4 | I have tried coated strings in the past ( incl... |
| **10259** | 4 | Well, MADE by Elixir and DEVELOPED with Taylor... |
| **10260** | 4 | These strings are really quite good, but I wou... |

10261 rows × 2 columns

In [8]:

dataset=df.filter(['reviewText'],axis=1)

df

Out[8]:

|  | **reviewerID** | **asin** | **reviewerName** | **helpful** | **reviewText** | **overall** | **summary** | **unixReviewTime** | **reviewTime** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | A2IBPI20UZIR0U | 1384719342 | cassandra tu "Yeah, well, that's just like, u... | [0, 0] | Not much to write about here, but it does exac... | 5 | good | 1393545600 | 02 28, 2014 |
| **1** | A14VAT5EAX3D9S | 1384719342 | Jake | [13, 14] | The product does exactly as it should and is q... | 5 | Jake | 1363392000 | 03 16, 2013 |
| **2** | A195EZSQDW3E21 | 1384719342 | Rick Bennette "Rick Bennette" | [1, 1] | The primary job of this device is to block the... | 5 | It Does The Job Well | 1377648000 | 08 28, 2013 |
| **3** | A2C00NNG1ZQQG2 | 1384719342 | RustyBill "Sunday Rocker" | [0, 0] | Nice windscreen protects my MXL mic and preven... | 5 | GOOD WINDSCREEN FOR THE MONEY | 1392336000 | 02 14, 2014 |
| **4** | A94QU4C90B1AX | 1384719342 | SEAN MASLANKA | [0, 0] | This pop filter is great. It looks and perform... | 5 | No more pops when I record my vocals. | 1392940800 | 02 21, 2014 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **10256** | A14B2YH83ZXMPP | B00JBIVXGC | Lonnie M. Adams | [0, 0] | Great, just as expected. Thank to all. | 5 | Five Stars | 1405814400 | 07 20, 2014 |
| **10257** | A1RPTVW5VEOSI | B00JBIVXGC | Michael J. Edelman | [0, 0] | I've been thinking about trying the Nanoweb st... | 5 | Long life, and for some players, a good econom... | 1404259200 | 07 2, 2014 |
| **10258** | AWCJ12KBO5VII | B00JBIVXGC | Michael L. Knapp | [0, 0] | I have tried coated strings in the past ( incl... | 4 | Good for coated. | 1405987200 | 07 22, 2014 |
| **10259** | A2Z7S8B5U4PAKJ | B00JBIVXGC | Rick Langdon "Scriptor" | [0, 0] | Well, MADE by Elixir and DEVELOPED with Taylor... | 4 | Taylor Made | 1404172800 | 07 1, 2014 |
| **10260** | A2WA8TDCTGUADI | B00JBIVXGC | TheTerrorBeyond | [0, 0] | These strings are really quite good, but I wou... | 4 | These strings are really quite good, but I wou... | 1405468800 | 07 16, 2014 |

10261 rows × 9 columns

In [9]:

dataset

Out[9]:

|  | **reviewText** |
| --- | --- |
| **0** | Not much to write about here, but it does exac... |
| **1** | The product does exactly as it should and is q... |
| **2** | The primary job of this device is to block the... |
| **3** | Nice windscreen protects my MXL mic and preven... |
| **4** | This pop filter is great. It looks and perform... |
| **...** | ... |
| **10256** | Great, just as expected. Thank to all. |
| **10257** | I've been thinking about trying the Nanoweb st... |
| **10258** | I have tried coated strings in the past ( incl... |
| **10259** | Well, MADE by Elixir and DEVELOPED with Taylor... |
| **10260** | These strings are really quite good, but I wou... |

10261 rows × 1 columns

In [10]:

**from** **nltk.sentiment.vader** **import** SentimentIntensityAnalyzer

In [11]:

sentences = dataset

In [12]:

sid = SentimentIntensityAnalyzer()

**for** sentence **in** sentences:

print(sentence)

ss = sid.polarity\_scores(sentence)

**for** k **in** sorted(ss):

print('**{0}**: **{1}**, '.format(k, ss[k]), end='')

print()

reviewText

compound: 0.0, neg: 0.0, neu: 1.0, pos: 0.0,

In [24]:

*# make a list of sentences, separating the tokens by white space.*

sentence\_list = []

**for** line **in** sentfile:

sentence\_list.append(line.split())

In [28]:

*# read the grammar file - the nltk data function load will not reload*

*# the file unless you set the cache to be False*

amg = nltk.data.load('file:amazon\_grammar.cfg', cache=**False**)

In [29]:

*# create a recursive descent parser*

am\_parser = nltk.RecursiveDescentParser(amg)

In [30]:

*# for each sentence print it and its parse trees*

*# if the grammar cannot parse a sentence, sometimes it gives an error and*

*# sometimes it just goes on to the next sentence with no parse tree*

**for** sent **in** sentence\_list:

print(sent)

**for** tree **in** am\_parser.parse(sent):

print (tree)

['reviewText']

In [31]:

sentence\_list

Out[31]:

[['reviewText']]

In [ ]: